Buch, N. y Velastin, S.A. (2014). Local feature saliency classifier for real-time intrusion monitoring. *Optical Engineering*, 53(7), 073108.

DOI: https://doi.org/10.1117/1.OE.53.7.073108

Local feature saliency classifier for real-time intrusion monitoring

Norbert Buch[a] and Sergio A. Velastin[b,c,*]
[a]Kristl, Seibt & Co., Baiernstrasse 122a, Graz 8052, Austria
[b]Universidad de Santiago de Chile, Department of Informatic Engineering, Av Ecuador 3659, Santiago, Chile
[c]Universidad Carlos III de Madrid, Departmento de Informática, Av. Universidad Carlos III 22, Colmenarejo 28270, Madrid, Spain

Abstract. We propose a texture saliency classifier to detect people in a video frame by identifying salient texture regions. The image is classified into foreground and background in real time. No temporal image information is used during the classification. The system is used for the task of detecting people entering a sterile zone, which is a common scenario for visual surveillance. Testing is performed on the Imagery Library for Intelligent Detection Systems sterile zone benchmark dataset of the United Kingdom's Home Office. The basic classifier is extended by fusing its output with simple motion information, which significantly outperforms standard motion tracking. A lower detection time can be achieved by combining texture classification with Kalman filtering. The fusion approach running at 10 fps gives the highest result of $F1 = 0.92$ for the 24-h test dataset. The paper concludes with a detailed analysis of the computation time required for the different parts of the algorithm. [DOI: 10.1117/1.OE.53.7.073108]

Keywords: texture saliency; visual surveillance; people tracking; clustering; foreground classification; sterile zone; intrusion detection; pedestrian tracking; closed circuit television.

Paper 140434 received Mar. 26, 2014; revised manuscript received Jul. 2, 2014; accepted for publication Jul. 3, 2014; published online Jul. 28, 2014.

1 Introduction

The use and installation of cameras for visual surveillance and security applications are continuously growing. To resolve the bottleneck of limited operator time for watching many cameras, video analysis systems can offer automatic event detection. One example scenario is detecting people entering a sterile zone, which could be a fence along a railway line, warehouse perimeters, or similar. Such scenes typically contain a protected area with a physical barrier (e.g., fence) and a restricted (sterile) zone bordering the barrier. Sterile zones in those cases often contain greenery, gravel (railway), or other homogeneous surfaces. Typical camera installations provide images which approximately contain equal proportions of those two areas. We use the stringent testing framework given by the i-LIDS sterile zone test dataset of the United Kingdom's Home Office[1] for such applications. This dataset is associated with a formal process of benchmarking commercial automatic surveillance systems and contains a wide range of environmental conditions (spanning all seasons and all weathers) as well as intrusion situations (walking, crawling, running, rolling, etc.) in two camera views (Fig. 1). Although these camera views have a certain amount of overlapping that could be exploited to improve the performance, the video files are not synchronized and the i-LIDS benchmark specifies performance for each camera in isolation, therefore we have not utilized the fusion of multiple views. However, it would be interesting to see future work explore such an approach. The i-LIDS program was inspired by a government need [informed by closed circuit television (CCTV) users] to rank systems, so that those with an appropriate level of performance could be recommended to government departments such as police forces. At the same time, the i-LIDS dataset provides a common set of data that researchers can use to compare results, even though some of its definitions of what constitute true and false detections might seem arbitrary and even idiosyncratic. The main challenge for the academic and industrial communities in such scenarios is to demonstrate the robust operation over a wide range of environmental conditions. Those conditions include camera shake, illumination changes, auto iris (adaptive gain), rain, snow, wild animals, and so on. Therefore, while the detection of intrusion might appear to be a simple problem, doing it reliably with low false alarms and doing it appropriately for operational deployment (typically an $F1$ score of 0.85 or better, please see below for the definition of the $F1$ metric), is a major challenge.

The metrics used for evaluation are defined by (and hence constrained by) the i-LIDS challenge[1] for the corresponding dataset and are a de-facto industrial standard for benchmarking these types of surveillance systems. An event-based evaluation is defined, where alarms reported within a window of 10 s of ground truth events are considered true positives (TPs). This is a somewhat arbitrary specification by the i-LIDS benchmark, especially as it does not consider the speed (e.g., slow) or the location of an intruder. Later in the paper, the results are not only reported for the 10-s window, but also for a 20-s window, which is shown benefit the proposed intrusion detection for slowly moving people. Any alarms reported outside this window are false positives (FPs). A person who might cause a second alarm, e.g., due to a lost track, would also count as an FP. Any missed person causes a false negative (FN). Those measures define recall R and precision P as follows:

*Address all correspondence to: Sergio A. Velastin, E-mail: sergio.velastin@ieee.org

0091-3286/2014/$25.00

Fig. 1 Examples from using the i-LIDS dataset. Correctly detected intrusions (True positives, TP) of the saliency classifier with motion extension for View 1 (top) and for View 2 (bottom). Note the snow in the middle image of View1 (top) and people crawling on the right.

$$R = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}, \tag{1}$$

$$P = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}. \tag{2}$$

i-LIDS[1] defines the $F1$ measure as a final metric, which combines recall R and precision P with recall bias α

$$F1 = \frac{\mathrm{RP}(1+\alpha)}{R + \alpha P}. \tag{3}$$

The recall bias α can have two values depending on the expected role of the system. For a system that is expected to be used for online monitoring, $\alpha = 0.65$ so as to penalize FP detections which could distract operators. Systems for event recording use $\alpha = 0.75$ to stronger penalize missed intrusions.

Commonly used methods such as foreground/background separation (e.g., using mixtures of Gaussians applied to temporal pixel variations) have problems dealing with the conditions represented in the dataset. Those problems arise from the need to maintain a background model, which often assumes a static camera view. Such limitations might be overcome by operating on a still image basis which does not assume a constant background model. It is observed that in many cases, sterile zones contain greenery, gravel (railway), or other homogeneous surfaces in which intrusion takes place. Therefore, we pose the intrusion detection problem as one of the detecting "saliency," where saliency refers to a local (in this case, corresponding to the intruder) significant difference in local texture features.

This paper is an extension of that presented in Ref. 2, and here we give the detailed description of the algorithm to allow other researchers to replicate our results. The main contributions of this work include, first, the proposal of a novel saliency classifier for intrusion detection in still images. Salient objects are detected in real time based on spectral texture features of image regions. This means that people are detected due to their texture difference compared to their surrounding texture. We extend the basic classifier by fusing saliency and a simple interframe difference motion mask to improve the robustness. A second extension uses Kalman filtering and allows motion silhouettes to initialize tracks to reduce the detection time (this is particularly relevant to the i-LIDS benchmark that allows only up to 10 s to detect an intruder, no matter how slowly the intruder is moving; any slower detection is considered an FN). Second, although the i-LIDS sterile zone dataset is an important and stringent benchmark, few results have been published and we argue that the academic community has not yet convincingly demonstrated appropriate performance for what is seemingly a simple problem: reliably detecting intrusion into a clean area. i-LIDS represents a major effort of years of wide consultation with end users, researchers, and manufacturers followed by data gathering over a full year and painstaking annotation for realistic operational conditions, therefore, we cannot afford to ignore it. One of the only few works we have seen that reports full results on this dataset is that of Ref. 3, which reports an $F1$ of 0.4 (well below what can be operationally acceptable) and of 0.69 (still below operational requirements) when "the three sequences with the higher error contribution are removed from the dataset." More recently, the same team[4] reports an approach to deal with the problem of broken tracks ("tracklets"), but they do not report $F1$ scores consistent with the i-LIDS definition, choosing to use their own definition of an $F1$ "tracking" score. It is also not clear if the false detections they report follow the i-LIDS benchmark definition. This is the current state of performance of published results, and hence we feel there is a contribution in reporting an algorithm with a much higher performance for which we also provide a detailed runtime and complexity analysis. Consequently, because we are dealing with a government-backed benchmark dataset with a well-defined training system and testing methodology, what might appear to be a specific solution has the strength of robustness such that it can be operationally applied, something that not many published algorithms can reasonably claim. Although sometimes it aims for what might seem generic solutions, engineering history shows us that success is to be found when a solution fits the problem well.

The reminder of the paper is organized as follows: The next section discusses the relevant work. The saliency classifier is introduced in Sec. 3. Extensions to the classifier are introduced in Sec. 4. Section 5 describes the dataset and provides details on the framework including timing analysis.

Full results are provided in Sec. 6. Section 7 concludes the paper.

2 Related Work

Relevant literature can be divided into two categories: methods exploiting temporal consistency by modeling background and methods operating on single frames. The first methods are usually fast to compute, but robustness in realistic conditions is limited. The proposed solution belongs to the second group, which gains robustness by solving the harder problem of foreground reasoning when considering only single frames. Tracking can be used to exploit the temporal consistency for reidentifying previously detected objects in a new frame.

A common solution for utilizing temporal consistency is to generate a pixelwise background model to estimate a motion foreground and to perform the tracking. The background model can be a single Gaussian as in the OpenCV blobtracker.[5] A background model based on mode in the temporal histogram is given in Ref. 6. The disadvantage of using a histogram is the slow adaptation for a changed background when a high mode is established. The seminal papers of Stauffer and Grimson[7,8] present a mixture of a Gaussians background model per pixel to deal with multiple background illumination characteristics by trading off computational speed against memory size. This approach generally provides good results for outdoor scenes. Sheikh and Shah[9] consider a probabilistic approach to jointly model regions of pixels. This allows the local spatial structure to be considered in a Markov random field, while Monnet et al.[10] use principal component analysis and an autoregressive model to predict a dynamic scene. This work is extended by Culibrk et al.,[11] who estimate the stable texture regions. Periodically changing backgrounds are modeled in Ref. 12 to incorporate distractions like escalators into the background model. More recently, Chen et al.[13] have described a foreground detection method that takes into account global illumination changes and also contains a process of shadow removal. A model based on texture blocks is proposed in Ref. 14 and used for tracking in Ref. 15. Pixel- and block-based approaches are combined in Ref. 16 with a hierarchical method. A parametric background model consisting of pixel intensity minimum, maximum, and maximum change is used in Ref. 17 as input for a tracker with appearance modeling. The real-time algorithm can distinguish between body parts using the motion silhouette, providing that there is enough visual information available, but it requires a high-image resolution. In Ref. 18, a background removal method is proposed based on color invariance. However, the method is only evaluated with their own indoor dataset. An interesting effort is the background models challenge (BMC).[19] However, in the BMC, the main data are two sequences of 1500 synthetic (a street and a traffic roundabout) frames each plus a set of nine "real application videos" of which six are less than 3 min, one is of 22 min, and two (including an interesting one of a snowy car park) are each of around 1 h and 15 min. Although this is an interesting effort, it is difficult to expect that systems that might do well on BMC would necessarily do well in an operational environment. This fact is the main motivation of the i-LIDS dataset, including its 24 h (2,160,000 frames) of test video under a range of conditions captured in a year of outdoor video recordings.

Articulated person tracking is performed in Ref. 20 with the aim to extract body poses. Batch processing combines a whole person detector and body part detector with a computation time of 14 to 20 s per frame. A more generic tracking algorithm based on covariance for reidentification of objects in new frames is proposed in Ref. 21. Target objects have to be manually tagged in the first frame. In contrast, the i-LIDS sterile zone scenario primarily requires a reliable detection of intruders and does not specify tracking requirements.

All background modeling approaches are affected by camera shake or fast scene changes, which are typical for realistic conditions as outlined earlier. Detection on single frames may overcome those problems; however, it increases the difficulty of detection as there is no temporal information available. Regression trees are used in Ref. 22 to classify pixels into road and nonroad for vehicle-mounted cameras assuming known road and nonroad seed areas. This does not require an offline training phase but has additional input from a laser range scanner. Based on training and structure from motion, Sturgess et al.[23] propose a segmentation system using graph cuts for understanding a road scene. Texton, color, location, and histogram of oriented gradients (HOG) descriptors are used in a boosting framework. A review of invariant pattern features is given in Ref. 24, which is commonly used in classifying images and content-based retrieval. Shotton et al.'s work[25] uses Textons to segment a single image and to perform multiobject recognition based on initial training. The current trend is to use pedestrian detectors that do not depend on background removal. Simonnet et al.[26] have presented a review of the main methods especially applied to cluttered conditions. One of the most popular detectors is the HOG detector,[27] which scans an image to detect pedestrians based on blockwise gradient histograms. An interesting fast algorithm is described in Ref. 28 using a two-stage classifier based on the multiblock local binary pattern and the weighted region covariance matrix. These kinds of algorithms rely on learning (through training) the appearances of people (usually upright). The i-LIDS dataset would require many detectors applied in parallel to cover all the presented ways of movement (walk, crawl on knee, crawl on stomach, roll, etc.), which can never be assumed to be exhaustive. Our previous work[2] draws on the later concepts but does not require a training stage due to the exploitation of texture. In this way, camera shake, illumination changes, and similar issues discussed earlier (and which are part of the i-LIDS challenge[1] to test algorithms) do not affect the algorithm. In addition, the complexity and runtime are still low, which is a typical limitation of single frame detectors. In this paper, we provide more details about the algorithm and additional results including runtime analysis.

2.1 *Overall Approach*

We propose a saliency classifier for intrusion detection in still images. Intruders are detected because of their differing texture compared to the surrounding texture in the image. This is achieved through the analysis of the texture of local image patches in a video frame. To analyze the local texture, the input image is divided into patches from which spectral features are generated. To identify image areas with similar texture, the patches are clustered in spatial and feature spaces.

Comparing the texture features of those clusters gives an indication of homogeneity of the image or saliency, if some clusters' features significantly differ from the rest. Those differing clusters are likely to correspond to intruders and are labeled as foreground. In this way, clusters are evaluated for saliency within a single frame. Foreground detections (intruders) per frame are accumulated over time to build trajectories of object centers in image space, which are evaluated for an intrusion condition. Following the i-LIDS definition of an "alarm," an intrusion takes place if "a person is present in the detection zone" (the ground next to the fence). The detection approach itself does not rely on the temporal consistency of the frames. In this way, camera shake, illumination changes, and similar issues discussed earlier do not affect the algorithm. In addition, the complexity and runtime are still low and no training is required, which is a typical limitation of single frame detectors (e.g., Refs. 25 and 27).

In a second algorithm, we fuse information from the above classifier and a frame differencing mask. The combination adds robustness against appearance noise that mainly affects the saliency and also against shake/illumination changes that mainly affect the differencing mask. To decrease the alarm time of the system, we introduce a Kalman filter. The i-LIDS specification defines a hard (ad-hoc) time limit for alarms of 10 s after the first appearance of an intruder. By tracking partly visible people at the edge of the camera with a Kalman filter based on motion, this early evidence allows faster alarm triggers within the specified time.

3 Saliency Classifier

This section introduces a saliency classifier based on texture. Section 3.1 describes the five steps (top blocks in Fig. 2) of foreground estimation and demonstrates how the spectral features of image patches are used to detect salient foreground regions. Those regions are then combined into objects for which trajectories are build. Section 3.2 describes the framework to trigger intrusion alarms based on the trajectories (bottom blocks in Fig. 2).

3.1 *Foreground Estimation*

Potential intruding objects are estimated from nonhomogeneities of local texture features in a single image. The foreground is passed to the intrusion rule framework described in Sec. 3.2. The spatial distribution of features for local image patches in a single frame is analyzed for saliency. No temporal background information is accumulated. To do this, the analog input image is first divided into patches for which texture features are calculated. Refer to Fig. 2 for a block diagram; the five steps are discussed in more detail in the following sections.

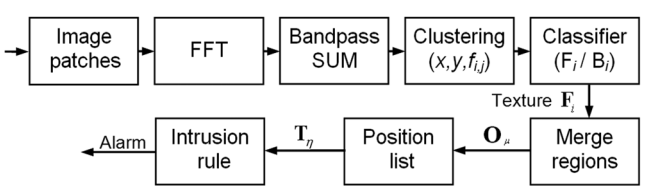

Fig. 2 Block diagram of the region saliency classifier.

3.1.1 *Image normalization and local patch generation*

First, the gray-level common intermediate format (360×288) representation of the analog video signal is used for foreground processing as the color information is only available during the daytime (this is typical for most CCTV installations). The monochrome input image is histogram stretched to ensure that the full dynamic range (intensity from e.g., 0 to 255) of the image is used under all lighting conditions.

The original shape of the histogram is preserved during this transformation. This early normalization increases the signal strength (but also the noise), which will be important to produce a consistent foreground under different lighting conditions.

The i-LIDS sterile zone dataset[1] specifies two regions $\mathbf{R}_i$, (i is either ground or fence), which can be represented by binary masks (please refer to Fig. 3, top left for an example image). Image patches are constructed for these regions and sequentially numbered with index j. Those patches $\mathbf{P}_{i,j}$ are 16×16 pixels and have a 20% overlap between them. The patches are fitted to the region mask $\mathbf{R}_i$ beginning from the top left to the bottom right. If the boundary of the region is not vertical, this will produce an unaligned grid of patches, as a new row of patches always starts at the edge of the region mask $\mathbf{R}_i$. The patch size is chosen as a power of 2 to enable the use of the fast Fourier transform (FFT). The size chosen should be as small as possible to allow for a fine foreground resolution but large enough to have sufficient texture information to discriminate between object and background at all distances to the camera. In practice, the minimum recommended size is 8×8.

3.1.2 *Fourier transform of individual patches*

To capture the texture and generate features for the image patches $\mathbf{P}_{i,j}$, the FFT is performed on each patch producing a spectrum $\tilde{\mathbf{P}}_{i,j} = \mathrm{FFT}(\mathbf{P}_{i,j})$ for each patch $\mathbf{P}_{i,j}$. The center of the spectral image $\tilde{\mathbf{P}}_{i,j}$ corresponds to the highest frequency, whereas the border corresponds to the lowest frequency. Noise is removed from the spectrum in the next step.

3.1.3 *Noise filter and feature generation*

The spectral patches $\tilde{\mathbf{P}}_{i,j}$ contain noise which would distract foreground detection. Low and high frequency components are removed from the spectra $\tilde{\mathbf{P}}_{i,j}$ providing the filtered spectrum $\hat{\mathbf{P}}_{i,j}$. Low frequencies (below index 2 in the spectral patches) contain the illumination conditions of the patch, which can significantly differ during the night, e.g., Fig. 1 on right. The average brightness of a patch determines the direct current component (index 1), and illumination gradients contain only low frequencies. High frequencies (above index 4) contain noise from the analog video feed and details, which are not discriminative for people entering the camera view. This leaves the mid-range frequency coefficients $2 \leq f_x \leq 4$ and $2 \leq f_y \leq 4$ (where f_x and f_y are the frequencies in the horizontal and vertical directions, respectively) for further processing. Spatial structures and details, which are smaller than people, are encoded. The random analog noise (thermal noise) causes spatially small distortions also called snow.[29] This is introduced by the video player, cables, and capture card. The performance has been observed to be robust against changing the filter frequency band by a one pixel index.

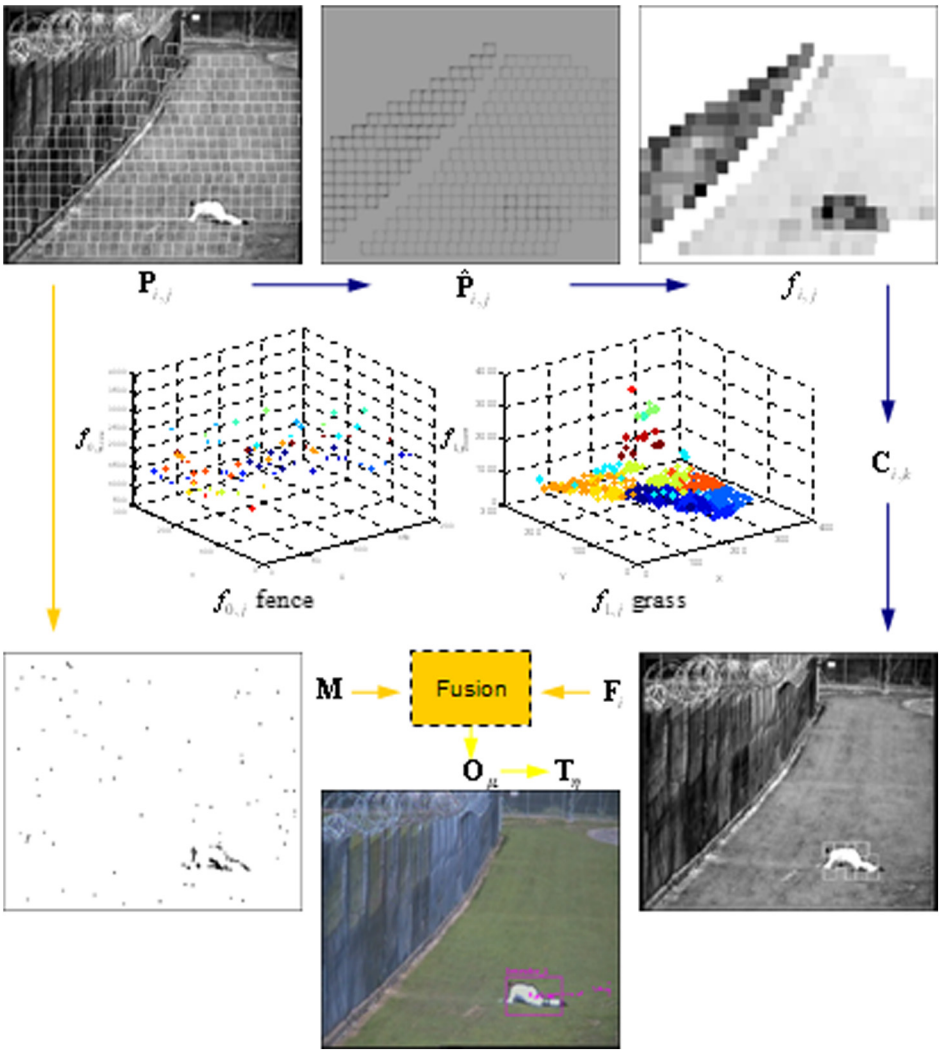

Fig. 3 Classification and detection process with illustrative intermediate images. The first row shows the input image with patches $\mathbf{P}_{i,j}$ highlighted. The middle image represents the spectral features $\hat{\mathbf{P}}_{i,j}$ of the patches with the last image depicting the features $f_{i,j}$ of every patch. For display purpose, the range of feature values is normalized to the full grayscale range in both regions. The second row shows the clustered patches with coordinates and feature value. On the left for the fence, all clusters (indicated by colors) are in a similar height range. In comparison, there are salient clusters for the person significantly above the background of the grass area. The last row on the right shows the detected foreground patches $\mathbf{F}_i$. The basic classifier uses this foreground only. Extensions to this method use the interframe motion mask **M** as on the bottom left and perform data fusion to provide objects $\mathbf{O}_\mu$ and the final trajectories $\mathbf{T}_\eta$ depicted in the central image.

To generate a scalar feature $f_{i,j}$ for each filtered spectral image patch $\hat{\mathbf{P}}_{i,j}$, the sum of the remaining spectrum over each patch $\hat{\mathbf{P}}_{i,j}$ is calculated as

$$f_{i,j} = \sum_{fx,fy} \hat{\mathbf{P}}_{i,j}. \tag{4}$$

The feature $f_{i,j}$ discriminates people from the background due to their different appearance, while at the same time it gives a similar response over the whole background (Fig. 3) in typical sterile zone scenarios as defined by Ref. 1. This is in strong contrast to the aim of describing texture discriminately as in Ref. 24 or similar texture classification approaches. The scalar features are much faster to process in the next steps compared to the whole spectrum and still provide sufficient discrimination to solve the task. However, it would be interesting to see future work to see if a nonscalar feature derived from the spectrum could produce better results.

3.1.4 *Clustering in feature and image space*

After calculating the feature for each image patch in the last step, salient patches, i.e., patches containing intruders, have to be identified to identify possible alarms. To find larger salient regions in the image, patches $\hat{\mathbf{P}}_{i,j}$ are clustered with respect to their location in the image x, y and their feature $f_{i,j}$. We have not found it necessary to normalize the feature, but it might be worth looking into this in more detail in the future. By considering clusters rather than single patches, larger support for saliency is accumulated. In addition, whole objects or large object fragments are represented by clusters. For the clustering itself, a hierarchical cluster tree

is generated to find N clusters $\mathbf{C}_{i,k}$ with cluster index $k \in [1, N]$ and the region index i from earlier. The choice of value of parameter N is discussed later. The mean feature value $\bar{f}_{i,k}$ of a cluster is used to detect an intruder. Ward's linkage algorithm[30] is used to combine clusters in the tree, which effectively minimizes the square of the Euclidean distance between elements in the clusters. The clusters of the example frame are illustrated in Fig. 3 as dots with different colors, where the clusters with higher values (red and green) in the right graph correspond to the intruder. For every cluster $\mathbf{C}_{i,k}$, the mean feature $\bar{f}_{i,k}$ is calculated as

$$\bar{f}_{i,k} = \frac{\sum_{\mathbf{P}_{i,j} \in \mathbf{C}_{i,k}} f_{i,j}}{|\mathbf{P}_{i,j} \in \mathbf{C}_{i,k}|}. \tag{5}$$

3.1.5 *Classification into foreground and background*

The resulting clusters $\mathbf{C}_{i,k}$ with mean feature $\bar{f}_{i,k}$ are now classified into foreground $\mathbf{F}_i$ and background $\mathbf{B}_i$. It is important to emphasize that this classification is based on the local texture feature statistic of a single frame, rather than relying on temporal foreground/background separation which can fail, e.g., for slow-moving objects or under camera shake. It is assumed that most of the image contains background and only a maximum of M patches are foreground $\mathbf{F}_i$. This is a valid assumption for any typical sterile zone scenario, where a camera covers a large area with a limited number of people entering the scene. The user choice of values for the number of patches in the foreground M and the number of clusters N are scene dependent and indirectly reflect the scale and perspective of the camera view. Scene dependency is common in many visual surveillance algorithms and in an application such as unattended intrusion detection that overwhelmingly uses static cameras, our long experience in working with end users and manufacturers indicates that users are satisfied with simple visual configuration procedures. In this case, these values can be obtained from considering the scene with the following procedure: The smallest foreground object to be detected (in this case a person) should approximately occupy one cluster, as this is the smallest unit to make a decision on intrusion. N, then, corresponds to the ratio between the number of image patches $\mathbf{P}_{i,j}$ for the smallest object to be detected and the total number of patches $\mathbf{P}_{i,j}$ in the region $\mathbf{R}_i$. For this dataset this gives $N = 15$. The number of foreground clusters M is calculated as the ratio between the smallest and largest object to be detected (please refer to Fig. 1 for examples of size variations). For this dataset this gives $M = 4$. Of course, it is possible to implement the semi-automatic ways of determining the values for these parameters (e.g., through machine learning), but this is outside the scope of this paper and we found that the simple process already proposed here results in a performance well above what has been reported for this benchmark dataset.

The concept of the potential foreground $\tilde{\mathbf{F}}_i$ is introduced as an initial foreground guess to allow the calculation of background statistics (single Gaussian) without contamination of foreground clusters. Potential foreground clusters are then evaluated against this background statistic to confirm them as final foreground. The potential foreground $\tilde{\mathbf{F}}_i$ contains M clusters with the highest mean feature $\bar{f}_{i,k}$ leaving all other clusters as background $\mathbf{B}_i = \{\mathbf{C}_{i,k} | \mathbf{C}_{i,k} \notin \tilde{\mathbf{F}}_i\}$. As background statistics, the mean features $\bar{f}_i$ of the background clusters and their variance σ_i^2 are calculated

$$\bar{f}_i = \text{mean}\{\bar{f}_{i,k} | \mathbf{C}_{i,k} \in \mathbf{B}_i\}, \tag{6}$$

$$\sigma_i^2 = \text{var}\{\bar{f}_{i,k} | \mathbf{C}_{i,k} \in \mathbf{B}_i\}. \tag{7}$$

The final foreground $\mathbf{F}_i$ consists of salient clusters of $\tilde{\mathbf{F}}_i$ fulfilling the saliency condition

$$\mathbf{F}_i = \{\mathbf{C}_{i,k} | \mathbf{C}_{i,k} \in \tilde{\mathbf{F}}_i \wedge \bar{f}_{ik} \geq T \cdot \sigma_i^2 + \bar{f}_i\}, \tag{8}$$

with saliency threshold $T = 5$. This implies that the foreground patches have to lie in the tail of the Gaussian background model and have higher feature values. The approach for foreground evaluation is similar to the mixture of Gaussians as used in Ref. 7 to model foreground and background. Their background threshold used here corresponds to the saliency threshold. The fact that background is not temporally modeled here requires the threshold to be applied to the feature value rather than the distribution proportion. The graphs in Fig. 3 show similar absolute values for both fence and grass clusters, but the clusters of the person in the grass are significantly elevated above the background clusters.

3.2 *Intrusion Rule Framework*

The intrusion rule framework first generates objects from the foreground and then evaluates their trajectory for an intrusion condition. This is illustrated by the bottom blocks in Fig. 2. First, spatially close foreground clusters are merged into single objects $\mathbf{O}_\mu$ with index μ. This means that clusters with patches overlapping each other are merged. Large objects close to the camera are usually segmented with several clusters due to the camera perspective as discussed above.

The positions of objects $\mathbf{O}_\mu$ are logged over time in image space to generate trajectories $\mathbf{T}_\eta$ with index η. Those trajectories are analyzed to detect genuine intrusions. Objects are associated with the closest trajectory based on the Euclidean distance in image coordinates. This is sufficient due to the low false detection rate of the saliency classifier and a typical low number of trajectories. If there is more than one object in a frame, multiple trajectories are generated or updated. This simple accumulation of positions will be extended by a Kalman filter in Sec. 4.2. However, this commonly used method helps in reducing the alarm time, but results in poorer performance, as will be shown in the results section, due to the simplistic assumption of constant velocity and a zero-mean Gaussian acceleration. This illustrates that the use of a Kalman filter is not appropriate to all tracking problems, perhaps something that is not always appreciated in this field. All trajectories $\mathbf{T}_\eta$ are considered for an alarm condition. The alarm rule requires a trajectory to have accumulated support from the saliency classifier for 2 s (this arises from the benchmarking requirements in i-LIDS) and the horizontal motion component has to be consistently toward the fence (i.e., left or right, depending on the side of the fence). A longer time window would increase the performance due to the increased evidence of an intruder; however, the stringent time window defined by the i-LIDS specification requires quickly raising alarms. The fence location (left or

right) is obtained from the i-LIDS scenario definition together with the sterile zone masks. An example frame with an intruder and trajectory is shown in Fig. 3.

4 Extensions to the Classifier

Two extensions are proposed for the saliency classifier. The first incorporates simple inter frame difference motion estimation to reduce the false detections by information fusion. The second introduces a Kalman filter to improve the trajectory quality and shorten the alarm triggering time. The information fusion resulted in a significant performance increase, whereas the extension with Kalman improved the time, but degraded performance in general.

4.1 *Motion Extension*

The algorithm described in Sec. 3 does not use any temporal information for detecting objects. The main reason for false detections is the existence of objects in an image. Examples of those would be fence shadows, small clouds, etc., which are stationary. The algorithm can be improved by incorporating motion information and fusing the information with the result of texture analysis introduced earlier. The motion extension incorporates temporal information by interframe differencing for motion foreground estimation, see Fig. 4 for a block diagram (orange color). A dynamic threshold is applied to the absolute frame difference of two consecutive frames, so that 10% of pixels are selected as foreground. This is implemented by sorting all pixels according to gray-level values of a frame and then taking the brightest 10% as foreground (those pixels with the biggest difference to the previous frame). Considering the size assumptions for objects from Sec. 3.1, this enforces that only part of a frame (e.g., a person if present) can be foreground at any given time.

In this way, significant global changes in image conditions (e.g., illumination change due to sun) can be dealt with and only the most significant moving objects are selected. If there are no moving objects, the foreground evenly represents the distributed small noise pixels, which are not considered for further processing. If a person were to remain stationary for some time, they would be picked again as they start to move. A morphological opening with a 3×3 kernel is applied to eliminate the small noise and join up larger regions to result in the final motion mask $\mathbf{M}$.

4.1.1 *Information fusion*

On the one hand, motion information is affected by camera shake, fast changing illumination conditions, etc., which is typical for this application as pointed out earlier. On the other hand, it is robust against the existence of stationary objects, which could affect only the saliency classifier. The information fusion requires valid objects to have support from saliency detection and (at the same time) motion pixels in the bounding box of the saliency detection. In this way, an object requires detection from both algorithms. The saliency bounding box is typically oversized due to the coarse structure of image patches, and, therefore, comfortably encloses the corresponding motion pixels. The number of motion pixels required was chosen as low as possible to avoid rejection of slowly moving intruders, but larger than the typical number of noise pixels in texture bounding boxes. The fusion reduces false detections as noise for appearance and motion is independent and, therefore, less likely to occur jointly. This allows lower detection thresholds for both detectors, which significantly reduce FNs (missed intrusions) by simultaneously increasing FPs (ambiguous alarms) of both classifiers. The fusion of both algorithms eliminates those additional FPs and avoids an overall increase.

4.2 *Kalman Filter Extension*

The algorithms proposed here so far suffer from a delay until the first detection of a person (i.e., latency). Very slow moving people stay partly occluded by the edge of the camera for a significant time, which potentially delays detection. The second extension with a Kalman filter overcomes this problem by allowing tracks to be initialized purely by small motion regions. This motion estimation is very noisy. In contrast to the basic intrusion detection system, some filtering is required to provide the consistency for trajectories. Please refer to Fig. 5 for a block diagram and to Fig. 6 for visual results. Examples in Fig. 7 show people who may stay partly occluded until the latest possible alarm triggering time. The trajectory generation is now performed by a Kalman filter with a constant velocity model. First, silhouettes $\mathbf{S}$ are extracted from the motion mask as connected components. This allows salient objects $\mathbf{O}_\mu$ as well as silhouettes $\mathbf{S}$ to update tracks, but alarms still require saliency detection in addition to silhouettes at some point of a trajectory.

New tracks are initialized for both of those inputs. Allowing silhouettes $\mathbf{S}$ to initialize trajectories $\mathbf{T}_\eta$ requires silhouettes of minimum size τ pixels to eliminate the analog video noise discussed above. Trajectories contain a sequence of object locations (x, y) over time, where the centroid of a silhouette $\mathbf{S}$ becomes the first object location in the trajectory. In comparison, the saliency classifier has a much higher precision, and in practice does not require a minimum size

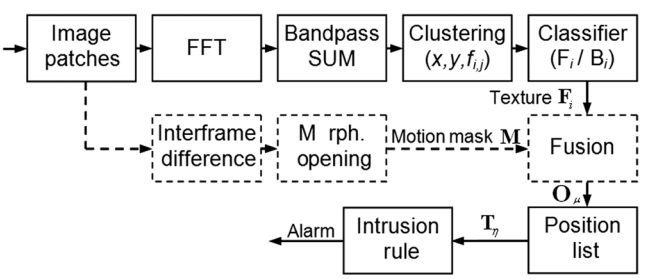

Fig. 4 Block diagram for the saliency classifier with motion extension.

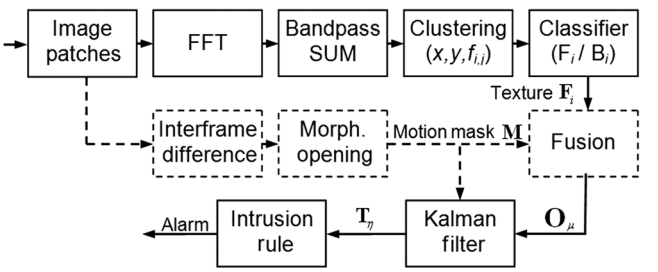

Fig. 5 Block diagram for saliency classifier with Kalman filter extension.

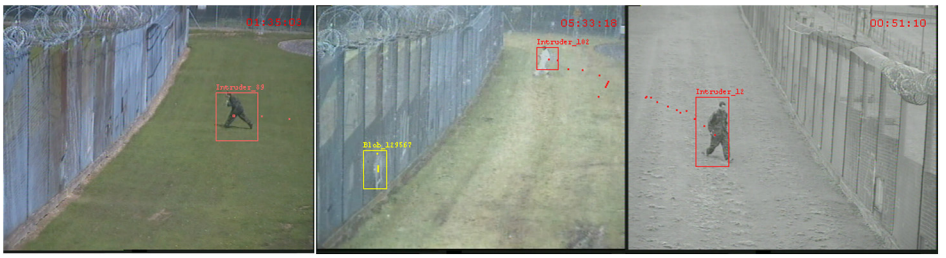

Fig. 6 TP examples of Kalman extension showing smooth tracks. Note the person rolling sideways in the image on the left, which indicates the various ways the fence is approached in the i-LIDS dataset.

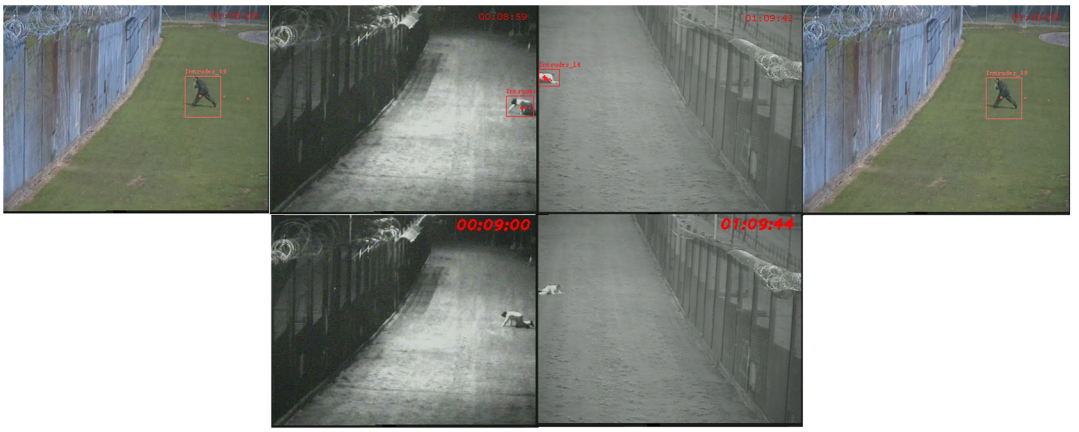

Fig. 7 Comparison of alarm triggering time. The top row shows the frame when the system with Kalman filter triggered an alarm. The bottom row shows later alarms of the system without the filter, especially when intruders are partly occluded by the edge of the camera for a long time.

filter. All trajectories $\mathbf{T}_\eta$ have an associated Kalman filter. To update those filters, a measurement $\mathbf{z} = (x_m, y_m)$ of an object location is required. To associate trajectories and objects, the distance between a Kalman filter prediction $(\hat{x}, \hat{y})$ and object locations is evaluated. The closest object is used for the update according to Eq. (9). Positions of salient objects are denoted (x_o, y_o) and for silhouettes (x_s, y_s), which defines the measurement selection as

$$\mathbf{z} = \begin{cases} (x_o, y_o) & \text{if } \min\left[\sqrt{(\hat{x}-x_o)^2 + (\hat{y}-y_o)^2}\right] < \min\left[\sqrt{(\hat{x}-x_s)^2 + (\hat{y}-y_s)^2}\right] \\ (x_s, y_s) & \text{else} \end{cases}. \tag{9}$$

The update with the silhouettes $\mathbf{S}$ allows trajectories $\mathbf{T}_\eta$ to start at the first appearance of a person at the edge of the camera and to fill temporal gaps in the saliency detection. The alarm delay time is reduced by this early detection of partly occluded people before the saliency classifier triggers for the first time (see Fig. 7 and Sec. 6.2). Saliency detection is mandatory for an alarm to be raised only to overcome the limitations of motion-based systems discussed in Sec. 2.

5 i-LIDS Testing

The system is tested on the i-LIDS dataset, which imposes particular requirements for system design. A runtime analysis for the real-time performance of the system is provided.

5.1 *Data*

The i-LIDS datasets[1] are licensed by the UK Home Office for image research institutions and manufacturers. The i-LIDS challenge aims at providing a benchmark for whole surveillance systems, which are defined by end users of the technology. The fact that the problem definition and data is generated by users ensures relevance and applicability of tested systems. Each dataset comprises 24 h of video sequences (2,160,000 frames) under a range of realistic conditions. The dataset is limited in terms of number of views, however, as producing a new view with the same variation of conditions carries a significant cost. The data are ideal for evaluating and comparing algorithms in the computer vision community and there is a gradual increase in take-up. We use the sterile zone test dataset, which consists of two views (one color, one black and white) during day and night with various weather conditions (day, night, rain snow, fast moving shadows, etc.). The test requires that one alarm for every intrusion event is raised and that the response is compared with the provided ground truth. Each of the two camera views (View 1 and View 2) is split into a sequence with alarms (208 total) and a sequence without alarms but with various distractions (birds, rabbits, etc.) recorded over the duration of a whole year. Refer to Figs. 1, 5, and 6 for detection examples. It should be mentioned that these data differ

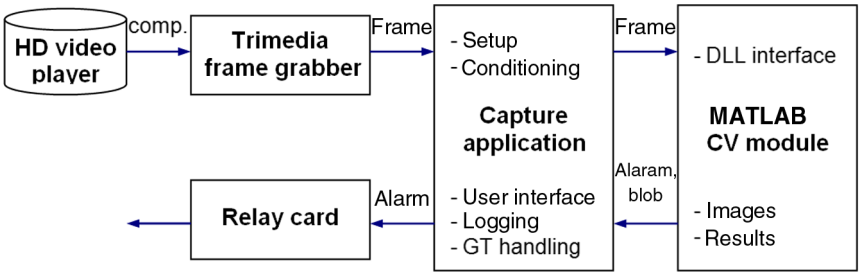

Fig. 8 Block diagram of system implementation with frame grabber, capture application and MATLAB computer vision module.

from typical detection and tracking evaluation, which usually contain many targets in relatively short video sequences. No objects are present for the majority of time in intrusion detection, but many distractions occur, which are captured in this dataset for practical relevance. This is realistic and stretches systems in terms of false alarms.

5.2 *Framework*

The system was designed according to i-LIDS requirements receiving an analog video input with 25 fps at PAL resolution and providing a relay alarm output (see Fig. 8). For our tests, the video was played back to the computer with a hard drive video player as a composite signal. An DSP-based frame grabber was used to sample the video and provide it to a capture application. The image processing is performed in a MATLAB® library, which is dynamically compiled and linked to the capture application. The capture application provides access to the hardware and performs conditioning of the input frames. Any brightness and contrast balance can be set up for the hardware to perform. This application also contains the user interface for ground truth handling and the setting up of experiments. The MATLAB module contains the algorithm described in this paper by taking frames as input and providing alarms and trajectories as outputs.

5.3 *Runtime Analysis*

The system is tested on a Pentium 4 with 2.4 GHz and 1GB RAM. A real-time performance of 9 to 10 fps can be achieved with an average processing time of 81 ms. Figure 9 shows the capture application's execution time over 200 processed frames. The overhead for the frame grabber is not shown. There is little overhead for performing the MATLAB call of 1.1 ms. The majority of time is spent for the patch analysis (FFT) and the subsequent clustering, classification, and information fusion. The Kalman filter takes a relatively small time (0.5 ms, i.e., 1.25% of the video frame period at 25 fps) for up to 30 tracks, but additional connected component analysis of the motion mask decreases the frame rate from 9 to 10 fps for the saliency classifier.

6 Results

This section describes the baseline algorithm and gives qualitative results with analysis.

6.1 *Baseline*

The baseline used is a standard Kalman filter blob tracker with a Gaussian background modeling based on the OpenCV library 5 blobtracker (parameters FG_1, BD_CC, CCMSPF, Kalman). The parameters of this tracker were set the same as in the proposed method, e.g., a minimum blob size filter was

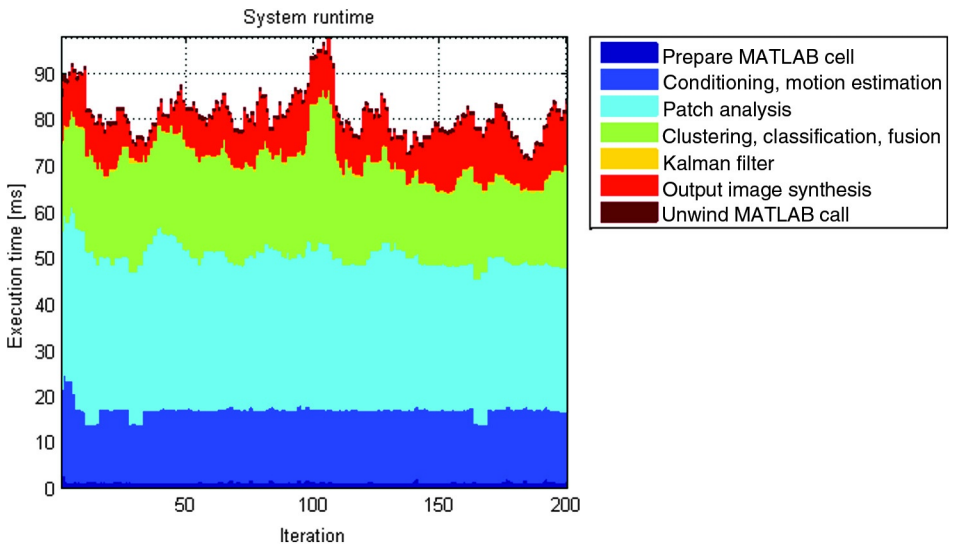

Fig. 9 Runtime analysis of the whole system implementation with average runtime of every module.

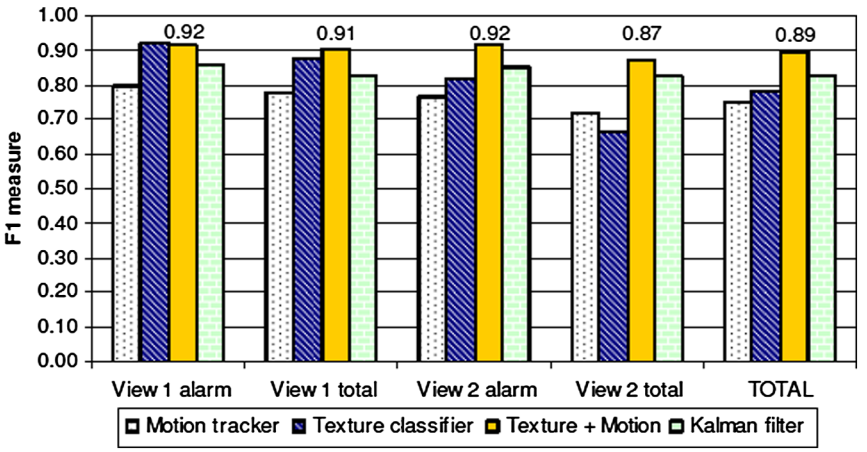

Fig. 10 Performance for 10-s alarm window. Results are shown for alarming sequences, total per view including the nonalarm sequences and total of the whole dataset.

correspondingly applied to the patch size from Sec. 3.1 to allow a fair comparison with the saliency classifier. Then, the intrusion rule framework from Sec. 3.2 is applied to the trajectories. This algorithm belongs to the first class mention in the related work section, which considers a stationary background and, in fact, already provides better results than those presented in Ref. 3; therefore, it is a valid baseline to which to compare. The main reasons for false detections are camera shake, fast illumination changes due to clouds, birds, and changes from the black and white color of the camera. This tracker is not without limitation, but it has been exposed to many applications and the behavior is well understood, so that the performance figures can be more easily interpreted.

6.2 *Analysis*

Four algorithms are compared in this section (see Fig. 10). The performance data are split into the two camera views (View 1 and View 2) and into sequences containing alarms and the total performance for the whole camera view. First, it is the baseline followed by the saliency classifier. The final two algorithms incorporate the motion extension and the Kalman filter into the saliency classifier. All performance values are for operation alert $\alpha = 0.65$ unless differently stated.

The baseline system achieves $F1 = 0.75$ in comparison to $F1 = 0.78$ of the saliency classifier. This outperforms the motion tracker; however, there are errors related to texture when shadows of the fences are detected. Low image contrast is the most common error cause and the reason for lower performance on View 2, see Fig. 11 for FPs from texture and FNs from low contrast. A high detection threshold is required to eliminate the FPs.

The saliency classifier with motion extension fuses information of those two approaches. It significantly outperforms both individual systems with $F1 = 0.89$ by exploiting the independence of the noise sources. A low threshold for saliency and motion detection allows the reduction of FNs from 35 to 17. To achieve this result, the saliency threshold was optimized resulting in $T = 2$, because lower thresholds produced arbitrary detection when no intruders were present in the image. With fusion, the FPs are also reduced from 44 to 16. One disadvantage of the fusion is the increased time to generate an alarm which sometimes extends past 10 s for slow moving people. Low image contrast remains the main reason for error as illustrated in Fig. 12. The lower result for "View 1 Alarm" compared to texture alone is due to a very slow moving person, which was not recognized through motion.

This increased alarm time inspired the second extension by using Kalman filtering and initializing tracks from motion silhouettes **S** in the interframe difference mask **M**. It is the last system shown in the figures. The minimum silhouette size is $\tau = 5$, which is larger than the typical noise observed in the data (e.g., Fig. 3).

The performance of the Kalman filter extension is lower compared to the motion extension. This is partly due to a larger number of FPs particularly during the snow sequence, but also because of its simplistic model as pointed out earlier.

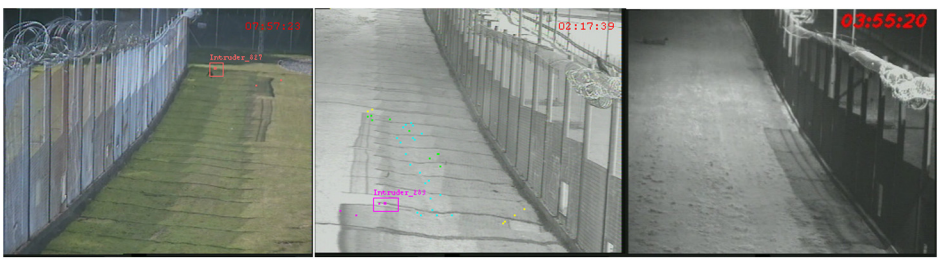

Fig. 11 (a) A wrongly detected bird flying toward the fence. (b) A false detection due to fast moving clouds present the same time as fence shadows, both errors are caused by texture. (c) A missed intruder due to low lighting conditions at night.

Fig. 12 Low contrast error examples. The first two images represent false negatives (FNs). The third image is a false positive (FP), where an alarmed track was lost. The intruder was later detected again and a second (false) alarm was raised.

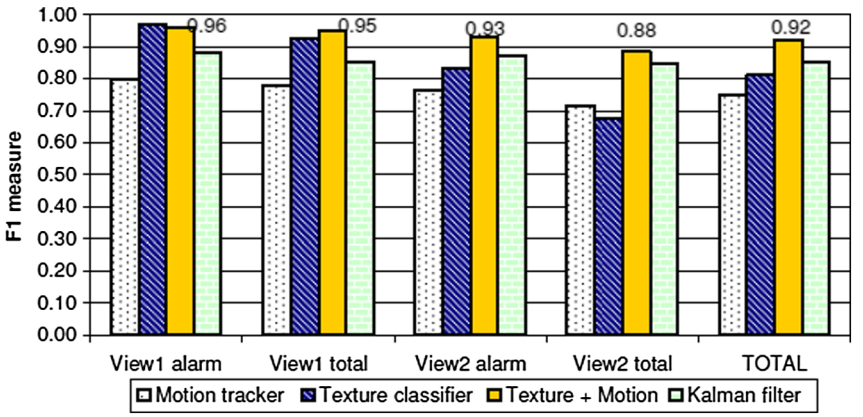

Fig. 13 Performance for 20-s alarm window. An improvement compared to 10 s is noticeable for both saliency classifiers due to later correct detections of slow moving people.

To keep those FPs down, the catch area for tracks is kept small, which causes some fragmented tracks for people. Those tracks are too short to alarm on which causes FNs. The average alarm time in a 10-s window is lowered from 3.4 s for motion extension to 3 s for Kalman filtering, which was the aim of the extension.

Both saliency classifier-based performance figures increase for a larger alarm window of 20 s (Figs. 13 and 14), which is caused by late correct detections. A late detection carries a high penalty according to the i-LIDS specification, as it is counted as FN and FP at the same time. As pointed out earlier, we have kept the evaluation consistent with the i-LIDS framework so that other researchers can compare results. However, from a user's perspective, a detection could be considered a TP, as long as the intruder is in the scene. A more realistic evaluation that could be used in the future would distinguish among "early detection" (as per the i-LIDS definition), "late detection" (beyond the i-LIDS threshold but while the intruder can be seen), and the normal FNs and FPs. The best overall performance is $F1 = 0.92$ for the saliency classifier with motion extension with the best performance for View 1 of $F1 = 0.95$. View 2 suffers from very low contrast, which is a particular problem for the saliency classifier; however, the motion extension significantly improves the performance by reducing the FPs from 30 to 7.

Finally, we compare the $F1$ measure for the two values of recall bias α. When using the event recording setting $\alpha = 0.75$ of i-LIDS,[1] the motion tracker performance is reduced by 0.3%. In contrast, the saliency classifier has increased the performance by 0.1%. The other two systems are not affected by α due to an even balance between FPs and FNs in the results.

	TP	FP	FN	F1
A1	85	17	28	0.80
T1		5		0.78
A2	62	10	33	0.76
T2		9		0.72

Motion Tracker 0.75

	TP	FP	FN	F1
A1	107	2	6	0.97
T1		9		0.92
A2	66	3	29	0.83
T2		30		0.68

Texture Classifier 0.81

	TP	FP	FN	F1
A1	108	4	5	0.96
T1		2		0.95
A2	83	3	12	0.93
T2		7		0.88

Texture + Motion 0.92

	TP	FP	FN	F1
A1	97	11	16	0.88
T1		7		0.85
A2	78	8	17	0.87
T2		4		0.85

Kalman Filter 0.85

Fig. 14 Detailed numbers of TP, FP, and FN with F1 measures for all four systems with alarm window setting of 20 s.

7 Conclusions

We proposed a texture saliency classifier to detect the objects in still images of the i-LIDS sterile zone dataset. Although the environment appears simple, this is stringent test covering operational conditions in a range of environmental situations. The system is implemented in C++ and MATLAB to operate in real time from an analog video input. This

approach overcomes the typical limitations of background modeling-based solutions. The runtime of 9 fps of the implementation is discussed in detail. The classifier outperforms the OpenCV blob tracker chosen as a baseline because it is widely available and others could verify our results. A first extension with information fusion between appearance and motion significantly increases the performance to $F1 = 0.92$ on the 24-h test dataset. The second extension using a Kalman filter is used to improve the alarm response times; however, it degrades the overall performance due to more FPs. The FPs are caused by noisy motion-based foreground estimation. Future work can focus on applying the classifier for moving camera platforms where no background estimation is possible, exploiting muticamera settings, comparing against other saliency features or people detectors (e.g., based on the popular histograms of gradients), and a parameter self-learning.

Acknowledgments

We are grateful to the Directorate of Traffic Operations at Transport for London for funding this work. We thank Fei Yin for providing the test results for the OpenCV blob tracker. Sergio A. Velastin gratefully acknowledges the stay at UC3M supported by the collaboration agreement "Chairs of Excellence" between University Carlos III and Banco Santander, and to the Chilean National Science and Technology Council (Conicyt) for its funding under grant CONICYT-Fondecyt Regular No. 1140209 ("OBSERVE").

References

1. Centre for Applied Science, and Technology (CAST), "Imagery library for intelligent detection systems," 26 March 2013, https://www.gov.uk/imagery-library-for-intelligent-detection-systems (25 March 2014).
2. N. Buch and S. A. Velastin, "Human intrusion detection using texture classification in real- time," in *First International Workshop on Tracking Humans for the Evaluation of Their Motion in Image Sequences THEMIS 2008*, J. Gonzàlez, T. Moeslund, and L. Wang, Eds., pp. 1–6, Universidad Autononoma de Barcelona (2008).
3. J. A. Vijverberg et al, "Two novel motion-based algorithms for surveillance video analysis on embedded platforms," *Proc. SPIE* **7724**, 77240I (2010).
4. J. A. Vijverberg, C. J. Koeleman, and P. H. N. de With, "Toward real-time and low-latency video object tracking by linking tracklets of incomplete detections," in *Proc. 10th IEEE Int. Conf. on Advanced Video and Signal Based Surveillance*, Krakow, Poland, pp. 300–305, IEEE, Piscataway, New Jersey (2013).
5. OpenCV, "Open source computer vision library," 5 February 2014, http://sourceforge.net/projects/opencvlibrary (25 March 2014).
6. J. Zheng et al, "Extracting roadway background image: mode-based approach," *J. Transp. Res. Board* **1944**(1), 82–88 (2006).
7. C. Stauffer and W. Grimson, "Adaptive background mixture models for real-time tracking," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 246–252, IEEE, Los Alamitos CA (1999).
8. C. Stauffer and W. Grimson, "Learning patterns of activity using real-time tracking," *IEEE Trans. Pattern Anal Mach. Intell.* **22**(8), 747–757 (2000).
9. Y. Sheikh and M. Shah, "Bayesian modeling of dynamic scenes for object detection," *IEEE Trans. Pattern Anal. Mach. Intell.* **27**(11), 1778–1792 (2005).
10. A. Monnet et al, "Background modeling and subtraction of dynamic scenes," in *Proc. 9th IEEE Int. Conf. on Computer Vision*, pp. 1305–1312, IEEE (2003).
11. D. Culibrk, B. Antic, and V. Crnojevic, "Real-time stable texture regions extraction for motion-based object segmentation," in *Proc. British Machine Vision Conf.*, British Machine Vision Association, London, UK (2009).
12. V. Leung et al, "Modelling periodic scene elements for visual surveillance," *IET Comput. Vis. J.* **2**(2), 88–98 (2008).
13. Z. Chen, T. Ellis, and S. A. Velastin, "Vehicle detection, tracking and classification in urban traffic," in *Proc. 15th IEEE Int. Conf. on Intelligent Transportation Systems (ITSC)*, Alaska, pp. 951–956, IEEE (2012).
14. M. Heikkila and M. Pietikainen, "A texture-based method for modeling the background and detecting moving objects," *IEEE Trans. Pattern Anal. Mach. Intell.* **28**(4), 657–662 (2006).
15. V. Takala and M. Pietikainen, "Multi-object tracking using color, texture and motion," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition, CVPR '07*, pp. 1–7, IEEE (2007).
16. Y. T. Chen et al, "Efficient hierarchical method for background subtraction," *Pattern Recognit.* **40**(10), 2706–2715 (2007).
17. I. Haritaoglu, D. Harwood, and L. Davis, "W4: real-time surveillance of people and their activities," *IEEE Trans. Pattern Anal. Mach. Intell.* **22**(8), 809–830 (2000).
18. H. Zhou, Y. Che, and R. Feng, "A novel background subtraction method based on color invariants," *Comput. Vis. Image Und.* **117**(11), 1589–1597 (2013).
19. A. Vacavant et al., "A benchmark dataset for foreground/background extraction," in *Proc. ACCV 2012, Workshop: Background Models Challenge*, pp. 291–300, LNCS 7728, Springer, Daejeon, Korea (2012).
20. D. Ramanan, D. Forsyth, and A. Zisserman, "Tracking people by learning their appearance," *IEEE Trans. Pattern Anal. Mach. Intell.* **29**(1), 65–81 (2007).
21. F. Porikli, O. Tuzel, and P. Meer, "Covariance tracking using model update based on lie algebra," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition, CVPR'06*, pp. 728–735, IEEE, Los Alamitos CA (2006).
22. B. Davies and R. Lienhart, "Using CART to segment road images," *Proc. SPIE* **6073**, 285–296 (2006).
23. P. Sturgess et al., "Combining appearance and structure from motion features for road scene understanding," in *Proc. British Machine Vision Conference*, London, UK (2009).
24. J. Zhang and T. Tan, "Brief review of invariant texture analysis methods," *Pattern Recogni.* **35**(3), 735–747 (2002).
25. J. Shotton et al, "Textonboost for image understanding: multi-class object recognition and segmentation by jointly modeling texture, layout, and context," *Int. J. Comput. Vis.* **81**(1), 2–23 (2009).
26. D. Simonnet et al, "Backgroundless detection of pedestrians in cluttered conditions based on monocular images: a review," *IET Comput. Vis.* **6**(6), 540–550 (2012).
27. N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition, CVPR'05*, pp. 886–893, IEEE, Los Alamitos CA (2005).
28. A. Halidou and Y. Xinge, "Fast pedestrian detection using BWLSD for ROI," in *Proc. 22nd Wireless and Optical Communication Conf. (WOCC)*, pp. 610–615, IEEE, Piscataway NJ (2013).
29. W. Ciciora, J. Farmer, and M. Adams, *Modern Cable Television Technology: Video, Voice, and Data Communications*, Morgan Kaufmann Publishers, San Francisco, CA (2004).
30. J. H. J. Ward, "Hierarchical grouping to optimize an objective function," *J. Am. Stat. Assoc.* **58**(301), 236–244 (1963).

Norbert Buch received his MSc degree in electrical engineering from the University of Technology, Graz (TUG), Austria, in 2006 and his PhD from Kingston University, UK, in 2010 for research in computer vision for traffic analysis. He is deputy head of the Software Department at Kristl, Seibt & Co., Austria, where he also leads the group for automation software. He received three excellence scholarships at TUG. He is member of the IET, IEEE, and OVE.

Sergio A. Velastin received his PhD from Manchester University in 1982 for work on computer vision. He is a research professor at the Universidad de Santiago de Chile and has worked in various EU-funded projects such as PRISMATICA, CARETAKER, AddPriv, ProtectRail, and LASIE. His current research interests include computer vision for pedestrian and traffic monitoring, and distributed visual surveillance systems. He is also a fellow of the IET and a senior member of the IEEE.